PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-002508
(43)Date of publication of application : 08.01.1993

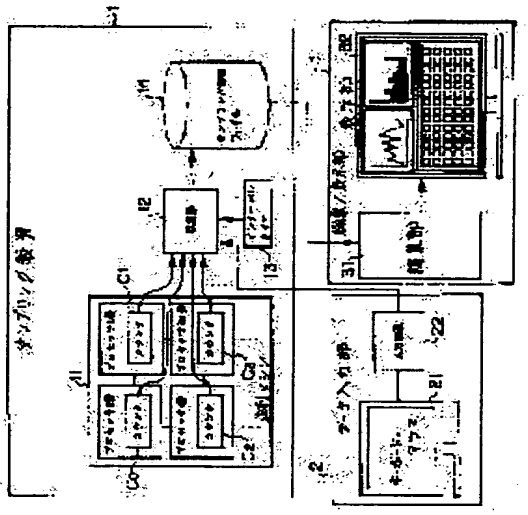
(51)Int.Cl. G06F 11/32
G06F 3/14
G06F 11/34

(21)Application number : 03-150673
(22)Date of filing : 21.06.1991

(71)Applicant : FUJITSU LTD
(72)Inventor : AIKAWA SEIICHI
KAMIKO MAYUMI
KUBO HIDEYUKI
MATSUZAWA FUMIKO

(54) PROGRAM OPERATION ANALYZER

(57)Abstract:
PURPOSE: To reduce cost and to attain efficient working in a program operation analyzer for supporting program development and evaluation/analysis in a supercomputer or a parallel machine constituted of plural processors.
CONSTITUTION: The parallel machine constituted of plural processors is provided with plural counters C0 to Cn included in respective processors to measure the execution frequency and interruption frequency of an instruction executed in each processor, an interval timer 13 for generating an interruption signal in each specified time interval, a collecting part 12 for collecting the contents of respective counters at the time of receiving the interruption signal, a sampling information file 14 for sampling collected data in the collecting part 12 and filing the sampled data, a data input part 2 for requesting the interruption of execution, and an edition/display part 3 for displaying the operating status of the program by means of a broken line graph, a bar graph, a pattern graph, and so on.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平5－2508

(43)公開日　平成５年(1993)１月８日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/32		Ａ　8725－5Ｂ		
3/14	３２０	Ａ　8725－5Ｂ		
11/34		Ｎ　8725－5Ｂ		

審査請求　未請求　請求項の数５(全 11 頁)

(21)出願番号　特願平3－150673

(22)出願日　平成３年(1991)６月21日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者　相川　聖一
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(72)発明者　神子　真弓
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(72)発明者　久保　秀行
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　大菅　義之　(外１名)

最終頁に続く

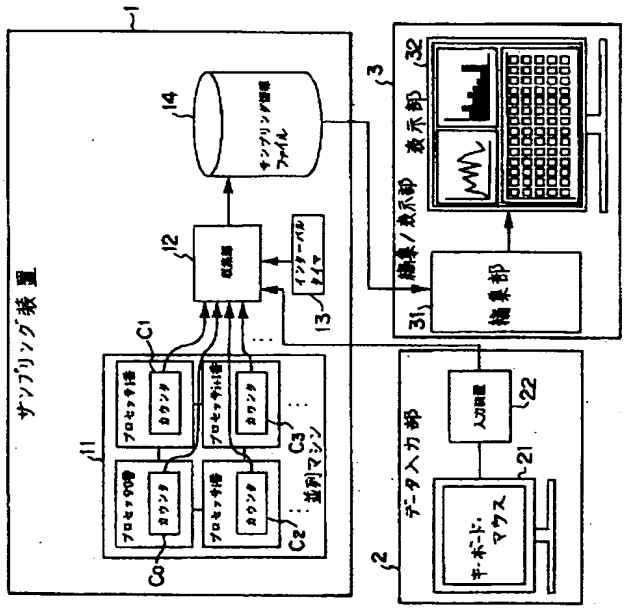
(54)【発明の名称】　プログラム動作解析装置

(57)【要約】　(修正有)

【目的】　スーパーコンピュータや複数のプロセッサから構成される並列マシン上でのプログラム開発やプログラムの評価・解析を支援するプログラム動作解析装置に関し、コストを削減し作業の効率化を図るものを提供する。

【構成】　複数のプロセッサから構成される並列マシンで、プロセッサごとに設けられ、各プロセッサで実行した命令の実行回数と中断回数を計測するカウンタC_0，C_1，・・・と、指定時間間隔ごとに割り込み信号発生するインターバルタイマ１３と、その割り込み信号を受けて、各カウンタの内容を収集する収集部１２と、収集部１２での収集データからサンプリングしてファイルするサンプリング情報ファイル１４と、実行中断を要求するデータ入力部２を備えた構成とする。またサンプリング情報ファイル１４をもとに、プログラムの動作状態を折れ線、棒、パターングラフなどを用いて表示する編集／表示部３を備える。

本発明原理を説明するためのシステム構成図



【特許請求の範囲】

【請求項１】　複数のプロセッサから構成される並列マシンにおいて、各プロセッサごとに設けられ、各プロセッサ内で実行した命令の実行回数と中断回数を計測するカウンタ（C_0，C_1，・・・）と、指定された時間間隔ごとに割り込み信号を発生するインターバルタイマ（１３）と、このインターバルタイマ（１３）からの割り込み信号を受けて、上記各カウンタ（C_0，C_1，・・・）の内容を収集する収集部（１２）と、この収集部（１２）で収集した情報をサンプリング情報としてファイルするサンプリング情報ファイル（１４）と、上記収集部（１２）に対し、状況に応じて割り込み信号を与えることのできるデータ入力部（２）と、上記サンプリング情報ファイル（１４）にファイルされた情報をもとにプログラムの動作状況を表すデータを作成し、そのデータを視覚的に表示する編集／表示部（３）と、を有することを特徴とするプログラム動作解析装置。

【請求項２】　上記データ入力部（２）から割り込み信号を入力することにより、プログラムの実行を中断させて、その時点までの上記各カウンタ（C_0，C_1，・・・）の内容を強制的に収集部（１２）に収集させて、サンプリング情報ファイル（１４）にファイルしたのち、編集／表示部（３）にて上記プログラムの動作状況を視覚的に表示するようにしたことを特徴とする請求項１記載のプログラム動作解析装置。

【請求項３】　プログラムの動作状況の視覚的な表示手段として、サンプリング情報ファイル（１４）のサンプリング情報から、時間の経過に伴うプログラム全体の実行回数または中断回数の変化を表すデータと、時間の経過に伴うプログラム中の各命令ごとの実行回数または中断回数の変化を表すデータをそれぞれ作成し、並列マシン上でのプログラム全体の実行状況および並列マシン上でのプログラム中の各命令ごとの実行状況をそれぞれグラフで表示することを特徴とする請求項１記載のプログラム動作解析装置。

【請求項４】　プログラムの動作状況の視覚的な表示手段として、サンプリング情報ファイル（１４）のサンプリング情報から、各プロセッサ上でのプログラム全体の総実行回数または総中断回数を表すデータと、各プロセッサ上でのプログラム中の各命令ごとの総実行回数または総中断回数を表すデータをそれぞれ作成し、並列マシン上でのプログラム全体の各プロセッサ間負荷バランスおよび並列マシン上でのプログラム中の各命令ごとの各プロセッサ間負荷バランスをそれぞれグラフで表示することを特徴とする請求項１記載のプログラム動作解析装置。

【請求項５】　プログラムの動作状況の視覚的な表示手段として、サンプリング情報ファイル（１４）のサンプリング情報から、時間の経過に伴う各プロセッサ上でのプログラム全体の実行回数または中断回数の変化を表すデータと、時間の経過に伴う各プロセッサ上でのプログラム中の各命令ごとの実行回数または中断回数の変化を表すデータをそれぞれ作成し、各プロセッサ上での経過に伴うプログラム全体の実行状況および各プロセッサ上での時間の経過に伴うプログラム中の各命令ごとの実行状況をそれぞれグラフで表示することを特徴とする請求項１記載のプログラム動作解析装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、スーパーコンピュータや複数のプロセッサから構成される並列マシン上でのプログラム開発やプログラムの評価・解析を支援するプログラム動作解析装置に関する。

【０００２】

【従来の技術】並列マシン上でのプログラムの開発は、分散実行可能な並列アルゴリズムの設計、プログラムの開発、並列マシン上におけるプログラムの実行という処理をプログラムが正しく動作するまで、あるいは性能評価の各作業を満足な計算速度が得られるまで、順次繰り返すことにより行われる。

【０００３】このプログラムの開発にあって、実際には、プログラム中のバグ（bug）の存在位置、性能の悪い箇所やその原因を探るための系統だった解析・評価手段がないため、試行錯誤を繰り返しながら作業を進めていくことになり、効率良くプログラムを開発するのは困難であった。このため、一連の作業を計算機によって支援し、開発者の負担を軽減する必要がある。

【０００４】科学技術計算の分野では、プログラム中のある特定のループが膨大な量のデータを処理する場合が多い。このため、スーパーコンピュータに代表される高速計算機では、コンパイラで自動的にプログラム中のループを、パイプライン方式などで、並列実行可能な形式に変換することによって、プログラムを高速に実行する方式を採用している。

【０００５】したがって、コンパイラの最適化方式の良否が、高速にプログラムを実行させるためのキーテクノロジーとなるため、コンパイラ方式の良否を適切に評価することが重要となる。

【０００６】しかし、現状では、予め設定した例題のテストデータを、スーパーコンピュータで実行し、計算に要した時間を計測することによって良否を判断している。このため、性能が悪い場合に、なぜ悪いのかを探る手段は、人手による机上シミュレーションに頼ることになり、性能の評価・解析作業は非常に時間のかかるものとなっていた。

【０００７】一方、複数のプロセッサから構成される並列マシンを利用する場合においても、計算速度を飛躍的に向上させることが主たる目的となる。この並列マシンでは、１台のプロセッサで構成される計算機に比べて、プロセッサの台数に比例して計算速度が向上することが

要求される。例えば、１台のプロセッサでプログラムを実行する場合に比べて、１０台のプロセッサを使用する場合は、計算速度も１０倍速くなることが要求される。

【０００８】このようにプロセッサの台数に応じて計算速度が向上することを、一般に台数効果と呼んでいる。台数効果を向上させるためには、複数のプロセッサにて同時に、効率良くプログラムを実行する必要がある。このためには、ある特定のプロセッサに負荷（計算量）が集中しないように、プログラム中の負荷を複数のプロセッサで並列に実行可能な単位に分割し、かつ各プロセッサに割り付ける負荷が均等になるように分散させる必要がある。

【０００９】しかし現実には、ある処理が終わるまで他の処理を実行しないようにする、いわゆる同期に代表されるようなプログラムの実行順序の依存関係や、プログラム間でのデータ構造の共有があるため、プログラムを並列実行可能な単位に分割することは難しく、また分割できる場合でも、プロセッサ間での負荷のばらつきがあるため、単純にプロセッサに割当てただけでは、各プロセッサを同時に、かつ効率良く実行させることは極めて困難であった。

【００１０】また、並列マシン上でのデバッグにおいては、複数のプロセッサでプログラムが並列に動作し、特に無限ループやデッドロックにより、プログラムの実行が正常に終了しない場合があるため、トレースなどによって実行状況を判定してバグの存在位置を特定するのは非常に困難な作業となっている。

【００１１】

【発明が解決しようとする課題】上記したように、並列マシン上でのプログラムの開発は、分散実行可能な並列アルゴリズムの設計、プログラムの開発、並列マシン上でのプログラムの実行というような処理をプログラムが正しく動作するまで、あるいは性能評価の各作業を満足な計算速度が得られるまで順次繰り返すことになる。しかし、実際には、バグの存在位置や性能が悪い箇所、原因を探るための系統だった解析・評価手段がないため、試行錯誤を繰り返しながら作業を進めていくことになり、効率良くプログラムを開発するのは極めて困難であった。

【００１２】本発明は、スーパーコンピュータや複数のプロセッサから構成される並列マシン上でのプログラムのデバッグ、評価・解析、テストを支援することによって、各作業に要す時間や人員などに対応するコストを削減するとともに各作業の効率化を図るプログラム動作解析装置を実現することを目的とする。

【００１３】

【課題を解決するための手段】図１は本発明によるプログラム動作解析装置のシステム構成図であり、これにより本発明原理を説明する。同図において、１は複数のプロセッサから構成される並列マシン上でのプログラムの実行情報を抽出するサンプリング装置であり、並列マシン１１に設けられた各プロセッサ（ここでは各々のプロセッサにプロセッサ０番，プロセッサ１番，プロセッサ２番・・・というような番号を付す）に対応して設けられたカウンタC_0，C_1，C_2・・・と、これら各カウンタC_0，C_1，C_2に保持された内容を、後述するインターバルタイマ１３からの収集要求割り込みを受けて収集する収集部１２と、この収集部１２に対して、予め指定された時間間隔ごとに収集要求の割り込みを発生するインターバルタイマ１３と、上記収集部１２で収集された情報を、サンプリング情報としてファイルするサンプリング情報ファイル１４とから構成されている。

【００１４】上記各カウンタC_0，C_1，C_2・・・は、各プロセッサごとの実行回数と中断回数をカウントするもので、図２に示すように、例えば「ａｄｄ」という命令が何回実行されたかという実行回数の累積値が２３０８８回、実行が何回中断したかという中断回数の累積値が１００３回というように、各命令ごとに実行回数の累積値と中断回数の累積値をカウントする。

【００１５】また、図１において、２はキーボードやマウスなどのデータ入力端末２１と入力装置２２から成るデータ入力部であり、オペレータの判断により状況に応じてプログラムの実行を中断させるための中断要求割り込みを、収集部１２に与えることができる。つまり、無限ループやデッドロックによってプログラムの実行が正常に終了しないようなプログラムに対し、そのプログラムの実行を中断させるための中断要求割り込みを入力することができる。この中断要求割り込みが出されると、その時点までに各カウンタC_0，C_1，C_2・・・が保持している内容（実行回数や中断回数）を、収集部１２が収集して、その収集した情報をサンプリング情報ファイル１４に送るようになっている。

【００１６】そして、上記サンプリング情報ファイル１４にファイルされた実行回数や中断回数のサンプリング情報は、編集／表示部３に送られ、各命令ごと及びプログラム全体の実行状況、各命令ごと及びプログラム全体の各プロセッサ間負荷バランス、各プロセッサ上での各命令ごと及びプログラム全体の実行状況として表すデータに編集されたのち、表示部３２にて折れ線グラフ、棒グラフ、パターングラフなどで表示されるようになっている。

【００１７】図３は上記サンプリング情報ファイル１４にファイルされるデータ形式を示すもので、そのデータ形式は、例えば同図に示すようにサイクル番号１におけるプロセッサ番号０，１，２，・・・、プロセッサ番号０における命令の名称「ａｄｄ」、「ｆｕｎｃ１」とそれぞれの実行回数と中断回数、プロセッサ番号１における命令の名称「ｓｕｂ」、「ｆｕｎｃ２」とそれぞれの実行回数と中断回数、というように各サイクル毎に各プロセッサの命令ごとの実行回数と中断回数が書き込まれ

た形式となっている。ここで、中断回数とは、例えば各プロセッサからデータが到着するまで、命令が待ち状態に入ったというようなものをいう。この中断回数が多いということはそのプログラムが適切でないことを意味している。

【００１８】ところで、上記各プロセッサでの実行回数及び中断回数のカウントは、具体的には図４のような手段にてカウントする。つまり、各プロセッサの処理系には、図４で示すように命令のつながった実行キュー４０と中断キュー４１を持っており、通常は、実行部４２にて実行キュー４０から命令を順次取り出して、その命令を実行するが、データに過不足があった場合は、中断キュー４１に移す。このとき、実行部４２から実行キュー４０に命令を取り出す回数を、実行回数カウント部４３でカウントしたものが実行回数としてカウントされ、中断キュー４１に処理を移したときの回数を中断回数カウント部４４でカウントしたものが中断回数としてカウントされるようになっている。

【００１９】

【作用】このような構成において、その処理手順を図５，図６のフローチャートを参照しながら説明する。図５はサンプリング装置１の情報収集アルゴリズムのフローチャートであり、図６は編集／表示部３における情報の編集／表示アルゴリズムのフローチャートである。まず図５によりサンプリング装置１における情報収集アルゴリズムについて説明する。

【００２０】並列マシン１１の各プロセッサに設けられたカウンタC_0，C_1，C_2・・・の内容を初期化（カウント値＝０）するとともにサイクル番号を初期化（サイクル番号＝１）する（処理Ｓ１）。そして、データ入力部２からの中断要求割り込みが発生したか（処理Ｓ２）、次に実行する命令があるか（処理Ｓ３）、インターバルタイマ１３から割り込みが発生したか（処理Ｓ４）の判断を行い、通常は、図中Ｒで示すループに沿った処理を行う。つまり、中断要求割り込みがなく、次に実行する命令があり、インターバルタイマ１３からの割り込みがない場合には、与えられた命令を実行し、実行した命令の名称に対応するカウンタの値に１を加算する（処理Ｓ５）という処理を繰り返す。

【００２１】この動作は各プロセッサ毎に行われ、データ入力部２からの中断要求割り込みが出されるか、実行する命令がなくなるか、あるいはインターバルタイマ１３から一定時間ごとに割り込みが発生するかのいずれかまで行われ、各プロセッサ毎に命令の名称に対する実行回数、中断回数が対応するカウンタにカウントされていく。

【００２２】そして今一定時間経過後、インターバルタイマ１３から割り込みが発生すると（処理Ｓ４）、収集部１２は、各プロセッサに対応して設けられた各カウンタC_0，C_1，C_2，・・・のその時点における内容（命令の名称、実行回数の累積値、中断回数の累積値）を収集し、その収集した情報にサイクル番号、プロセッサ番号を付加してサンプリング情報ファイル１４に出力する（処理Ｓ６）。つまり、サンプリング情報ファイル１４には図３で示すような内容の情報がファイルされる。

【００２３】その後、各カウンタC_0，C_1，C_2，・・・の内容を初期化（カウント値＝０）し、サイクル番号を＋１する（処理Ｓ７）。そして、再び次の命令を実行し、実行した命令の名称に対応するカウンタの値に１を加算するというループＲの処理を行う。

【００２４】一方、上記した処理中、無限ループやデッドロックによってプログラムの実行状況が正常でなくなった場合には、オペレータ入力部２から中断要求割り込みを与えることで、強制的に各プロセッサに設けられているカウンタの内容（命令の名称、実行回数、中断回数）を収集部１２で収集する。そして各カウンタの内容にサイクル番号、プロセッサ番号を付加してサンプリング情報ファイル１４に出力する（処理Ｓ８）。また実行する命令がなくなった場合（処理Ｓ３）も上記同様処理Ｓ８へ移る。

【００２５】このような処理を繰り返すことにより、サンプリング情報ファイル１４には、図３で示すような情報がファイルされる。次に上記したサンプリング情報ファイル１４のファイル内容を用いて、編集／表示部３により、編集／表示アルゴリズムの処理フローを図６を参照しながら説明する。図６において、プログラム全体および各命令ごとの実行状況を折れ線グラフで表すための編集／表示部３のアルゴリズムを処理フローＳ２０、またプログラム全体及び各命令ごとの負荷バランスを棒グラフで表すための編集／表示部３のアルゴリズムを処理フローＳ３０、さらに各プロセッサ上でのプログラム全体及び各命令ごとの実行状況をパターングラフで表すための編集／表示部３のアルゴリズムを処理フローＳ４０として示し、図示実線で示す処理フローは各命令ごとの処理、点線で示す処理フローはプログラム全体（全命令）の処理をそれぞれ示している。

【００２６】まず、各命令ごとの編集／表示アルゴリズムの処理フローについて説明する。この場合、サンプリング情報ファイル１４からは、各命令の種類（名称）をキーとして、その命令に対するデータを取り出す（処理Ｓ１１）。

【００２７】そして、折れ線グラフで表示する処理フローＳ２０では、まず、各サイクルごとに全プログラムにて実行した実行回数と中断回数を集計する（処理Ｓ２１）。次に横軸にサイクル番号、縦軸に実行回数または中断回数を表示し（処理Ｓ２２）、上記集計結果に基づいて、各命令ごとの実行状況を折れ線グラフで表示する（処理Ｓ２３）。

【００２８】また、棒グラフで表示する処理フローＳ３

０では、まず、各プロセッサごとに全サイクル内での総実行回数と総中断回数を集計する（処理Ｓ３１）。次に横軸にプロセッサ番号、縦軸に実行回数または中断回数を表示し（処理Ｓ３２）、上記集計結果に基づいて、各命令ごとの各プロセッサに対する負荷バランスを棒グラフで表示する（処理Ｓ３３）。

【００２９】さらに、パターングラフで表示する処理フローＳ４０では、まず、各サイクルごとに各プロセッサでの各命令の実行回数と中断回数を集計する（処理Ｓ４１）。次に横軸にサイクル番号、縦軸にプロセッサ番号をとり、色の濃淡で実行回数または中断回数（回数多い→濃、回数少ない→淡）を表示し（処理Ｓ４２）、上記集計結果に基づいて、各プロセッサ上での命令の実行状況をパターングラフで表示する（処理Ｓ４３）。

【００３０】一方、プログラム全体の編集／表示部３のアルゴリズムでは、サンプリング情報ファイル１４から、全命令に対するサンプリング情報を取り出す。そして、折れ線グラフで表示する処理フロー２０では、前記したように処理Ｓ２１，Ｓ２２を行い、各プロセッサ上でのプログラム全体の実行状況を折れ線グラフで表示する（処理Ｓ２４）。また、棒グラフで表示する処理フローＳ３０では、前記したように処理Ｓ３１，Ｓ３２を行い、プログラム全体とプログラムを実行したプロセッサとの対応関係、つまり各プロセッサ間の負荷バランスを棒グラフで表示する（処理Ｓ３４）。さらにパターングラフで表示する処理フローＳ４０では、前記したように処理Ｓ４１，Ｓ４２を行い、各プロセッサ上でのプログラム全体の実行状況をパターングラフで表示する（処理Ｓ４４）。

【００３１】以上のようにして、プログラム全体及び各命令ごとの実行状況、プログラム全体の負荷バランス及び各命令ごとの負荷バランス、各プロセッサ上でのプログラム全体の実行状況及び各プロセッサ上での各命令ごとの実行状況をそれぞれグラフで表示するようにしたので、並列マシン上でのプログラムの開発において、バグの存在位置や性能の悪い箇所、原因を探るための系統だった解析・評価手段として、また、無限ループやデッドロックしている箇所などを視覚的に判別でき、これにより、試行錯誤を繰り返しながら作業を進めていたプログラムの開発、デバッグ、評価・解析、テスト作業の効率化に大きく寄与することができる。

【００３２】

【実施例】次に本発明の一実施例を説明する。ここでは実施例として、前記した本発明の原理をもとに、各グラフの具体的な表示例について説明する。サンプリング装置１におけるデータ収集手順は前記したように図５のフローチャートの通りである。すなわち、各プロセッサが命令を実行し、その実行した命令の種類ごとに、各カウンタにて実行回数及び命令の待ち状態などによる中断回数をカウントする。そして収集部１２は一定時間ごとにインターバルタイマ１３からの割り込みを受けて、その割り込みを受けた時点までのカウント内容を収集し、収集した情報をサンプリング情報ファイル１４に出力する。また、各カウンタC_0，C_1，C_2，・・・から収集部１２への情報収集は、インターバルタイマ１３から一定時間ごとの割り込みによるものだけでなく、無限ループやデッドロックなどプログラムの実行状況が正常でなくなった場合に、オペレータがデータ入力部２から中断要求割り込みを入力することで、強制的にその時点までの情報を収集して、サンプリング情報ファイル１４に送出する。

【００３３】このようにして、サンプリング情報ファイル１４には、図３のような、サイクル番号、プロセッサ番号、命令の名称、実行回数、中断回数がファイルされる。このファイルされた情報をもとに編集／表示部３にて、図６のフローチャートの処理手順によりグラフ化を行う。

【００３４】以下に具体的なグラフの表示例を示すとともに、その表示例によりプログラムの正当性などの評価を行う例について説明する。まず、図６の処理フローＳ２０により、プログラム全体の実行状況及び各命令ごとの実行状況を折れ線グラフで表示する場合について説明する。

【００３５】各命令ごとの実行状況を折れ線グラフで表示する場合、各命令ごとにその命令の名称をキーとしてサンプリング情報ファイル１４からデータを取り出し、各サイクルごとに全プロセッサ内での実行回数と中断回数を集計する。そして、横軸に時間の経過に伴うサイクル番号１，２，・・・をとり、縦軸にこの場合、実行回数をとって表示する。例えば、図７（ａ）は命令ｉの実行状況を折れ線グラフで表示したものである。

【００３６】また、プログラム全体の実行状況を折れ線グラフで表示する場合、サンプリング情報ファイル１４から、全命令のサンプリング情報を取り出し、以下上記同様の処理手順により、横軸に時間の経過に伴うサイクル番号、縦軸に実行回数をとることにより、例えば図７（ｂ）のような折れ線グラフで表示される。

【００３７】この図７（ａ），（ｂ）の折れ線グラフを見ることにより、各命令の実行状況及びプログラム全体の実行状況が視覚的に即座に把握することができる。並列マシン上で動作可能なプログラミング言語では、プログラムの実行過程において、ある処理が終わるまで次の処理を行わないようにするために同期を取る機能を有している。この同期の取り方が適切でないと、プログラム全体の実行に大きく影響を与える。つまり、図７（ａ），（ｂ）の折れ線グラフにおいて、定期的に実行回数が低下する部分（図中、矢印Ａで示す）がある。これによりプログラム中に不適切な同期の取り方をしている箇所があることを分かる。

【００３８】そこで、各命令ごとに実行状況を表示して

みると、例えば図７（ｃ）のように命令ｊが、他の命令が同期処理のために実行回数の低下している箇所で、逆に実行回数が増加（図中、矢印Ｂで示す）していることが分かる。したがって、命令ｊの同期の取り方を修正することにより、プログラム全体の実行回数が極端に低下することを防ぐようにして、実行効率の改善を図れば良いということがわかる。

【００３９】また、プログラムの実行が無限にループに陥っている場合やデッドロックにより、実行が中断している場合は、実行の途中でデータ入力部２から中断要求の割り込みを入力し、その時点までに収集したサンプリング情報を用いて、各命令の実行状況を表示することにより、プログラム中のどの部分が無限ループやデッドロックに陥っているかを判別することができる。図７（ｄ）は命令ｋが無限ループに陥っている様子を示している。つまり、既に実行が終了しているはずの命令が、いつまでも終了していないことが図７（ｄ）のグラフの実行回数から認識することができる。

【００４０】次に各プロセッサに対する負荷バランスを棒グラフで表示する例について説明する。この処理手段は図６の処理フローＳ３０にて行われる。通常、科学技術計算では、ある特定のループのみが何万回も実行されるため、その命令が最適化されているか否かがプログラム全体の実行期間に与える影響が大きい。図８（ａ）は命令ｉの各プロセッサに対する実行回数、同図（ｂ）はプログラム全体の各プロセッサに対する実行回数をそれぞれ棒グラフで示したもので、これにより各プロセッサ間負荷バランスを知ることができる。図８（ａ）により、命令ｉの各プロセッサ間負荷バランスの変動の大きさが、同図（ｂ）のようにプログラム全体の各プロセッサ間負荷バランスに大きな影響を与えていることがわかる。したがって、命令ｉの各プロセッサ間の負荷バランスを同図（ｃ）のように各プロセッサにほぼ均等に分散させるよう改善することによって、プログラム全体の各プロセッサ間負荷バランスも同図（ｄ）のように改善することができる。

【００４１】次に各プロセッサ上での命令の実行状況とプログラム全体の実行状況をパターングラフで表示する例について説明する。この処理手順は図６の処理フローＳ４０にて行われる。このパターングラフにおいて使用するパターンの意味を図９（ａ）に示す。同図（ａ）よりわかるように、色の濃淡にて回数の多い少ないを表示し、ここでは色が濃いほど回数が多いことを示している。図９（ｂ）は各プロセッサ間の負荷バランスの悪い場合の表示例、同図（ｃ）は各プロセッサ間の負荷バランスの良い場合の表示例を示すものである。同図（ｂ）からわかるように、各プロセッサの負荷にばらつきがあり、かつ定期的に、あるサイクルにおいて色の淡いパターンが縦方向に表示されている。これは実行回数が極端に低下していることを示している。つまり、プログラム中に前記したような強い同期が発生していることを示している。

【００４２】そこで、各命令ごとにパターングラフで表示を行い、同図（ｄ）に示すように、全体の負荷バランスに影響を与えている命令ｉとプログラム全体の実行に対して強い同期処理を行っている命令ｊを捜し出すことがきる。これにより、命令ｉの負荷バランスを改善するとともに、同期処理を改善することによって、同図（ｂ）に示すように各プロセッサ上で負荷バランスが均等化され、しかも同期により極端に実行回数が低下することのない効率の良いプログラムを開発することができる。

【００４３】なお、上記実施例においては実行回数の変動をグラフとして表示するようにしたが、中断回数の変動をグラフとして表示するようにしても良いことは勿論である。

【００４４】

【発明の効果】本発明によれば、スーパーコンピュータや複数のプロセッサから構成される並列マシン上でのプログラムの実行過程をグラフで表示するようにしたので、無限ループやデッドロックしているプログラムのデバッグやプログラム中の性能の悪い箇所を視覚的に即座に把握することができ、これにより、従来では試行錯誤を繰り返しながら作業を進めていたプログラムの開発、デバッグ、評価・解析、テストなどの各作業をきわめて効率良く行うことができる。

【図面の簡単な説明】

【図１】本発明の原理を説明するためのシステム構成図である。

【図２】図１で示したカウンタC_0，C_1，C_2・・・のカウント内容を示す図である。

【図３】図１で示したサンプリング情報ファイル１４にファイルされるサンプリング情報を示す図である。

【図４】実行回数と中断回数のカウント手段を概略的に示す構成図である。

【図５】本発明原理によるサンプリング情報収集アルゴリズムの処理手順を示すフローチャートである。

【図６】本発明原理による編集／表示部３のアルゴリズムの処理手順を示すフローチャートである。

【図７】同実施例において、各命令の実行状況とプログラム全体の実行状況を折れ線グラフで表示した例を示す図で、同図（ａ）は命令ｉの実行状況、同図（ｂ）はプログラム全体の実行状況、同図（ｃ）は命令ｊの実行状況、同図（ｄ）は命令ｋの実行状況をそれぞれ示す図である。

【図８】同実施例において、各命令に対する各プロセッサ間の負荷バランスとプログラム全体に対するプロセッサ間の負荷バランスを棒グラフで表示した例を示す図で、同図（ａ）は命令ｉに対する各プロセッサ間の負荷バランス、同図（ｂ）はプログラム全体に対する各プロ

セッサ間の負荷バランス、同図（ｃ），（ｄ）は同図（ａ），（ｂ）の改善後の負荷バランスをそれぞれ示す図である。

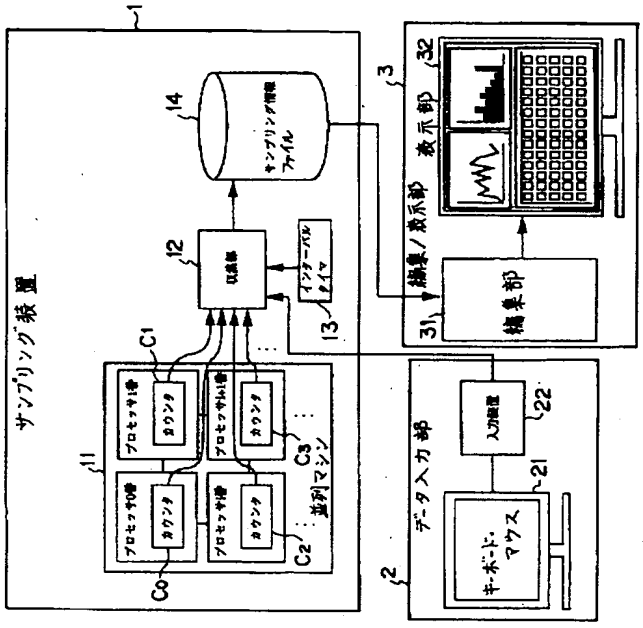
【図９】同実施例において、各プロセッサ上でのプログラムの実行状況をパターングラフで表示した例であり、同図（ａ）はパターングラフのパターンの色の濃淡と実行回数との関係を説明するための図、同図（ｂ）は負荷バランスの悪いプログラムの実行状況、同図（ｃ）は負荷バランスの良いプログラムの実行状況、同図（ｄ）は命令ｉに対する各プロセッサ上での実行状況、同図（ｅ）は命令ｊに対する各プロセッサ上での実行状況をそれぞれ示す図である。

【符号の説明】

１・・・・・サンプリング装置
２・・・・・データ入力部
３・・・・・編集／表示部
１１・・・・並列マシン
１２・・・・収集部
１３・・・・インターバルタイマ
１４・・・・サンプリング情報ファイル
C_0，C_1，C_2 ・・・カウンタ

【図１】

本発明原理を説明するためのシステム構成図



【図２】

各カウンタの内容の一例を示す図

命令の種類	実行回数	中断回数
add	23088	1003
sub		
⋮	⋮	⋮

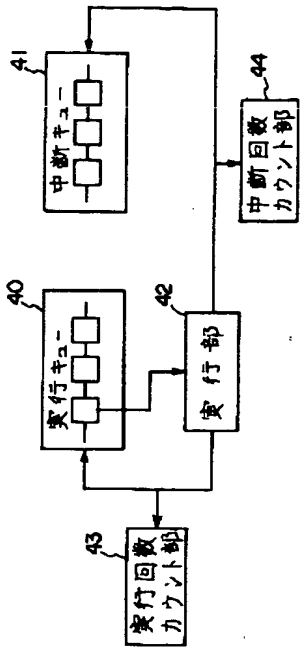
【図３】

サンプリング情報ファイルの内容を示す図

サイクル番号	プロセッサ番号	命令の名称	実行回数	中断回数
1	0	add	302	20
		func1	405	34
	1	sub	506	390
		func2	243	145
	2	func3	208	45
	⋮	⋮	⋮	⋮
2	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮

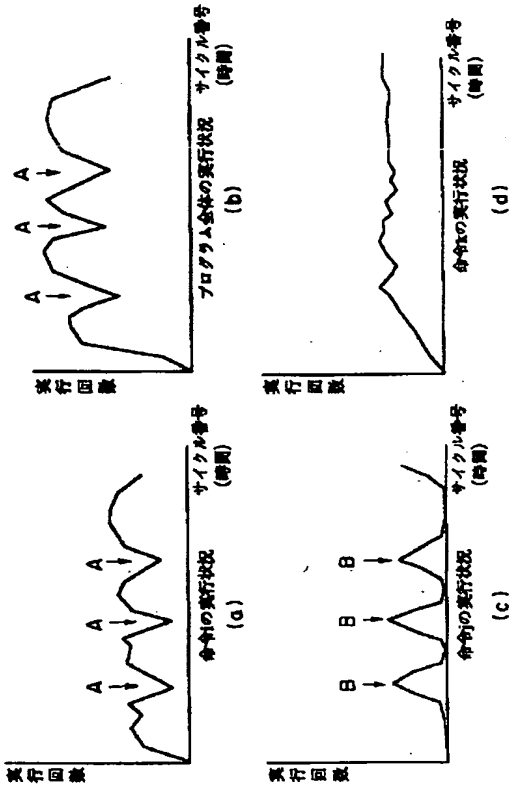
【図４】

実行回数と中断回数のカウント手段を示す図



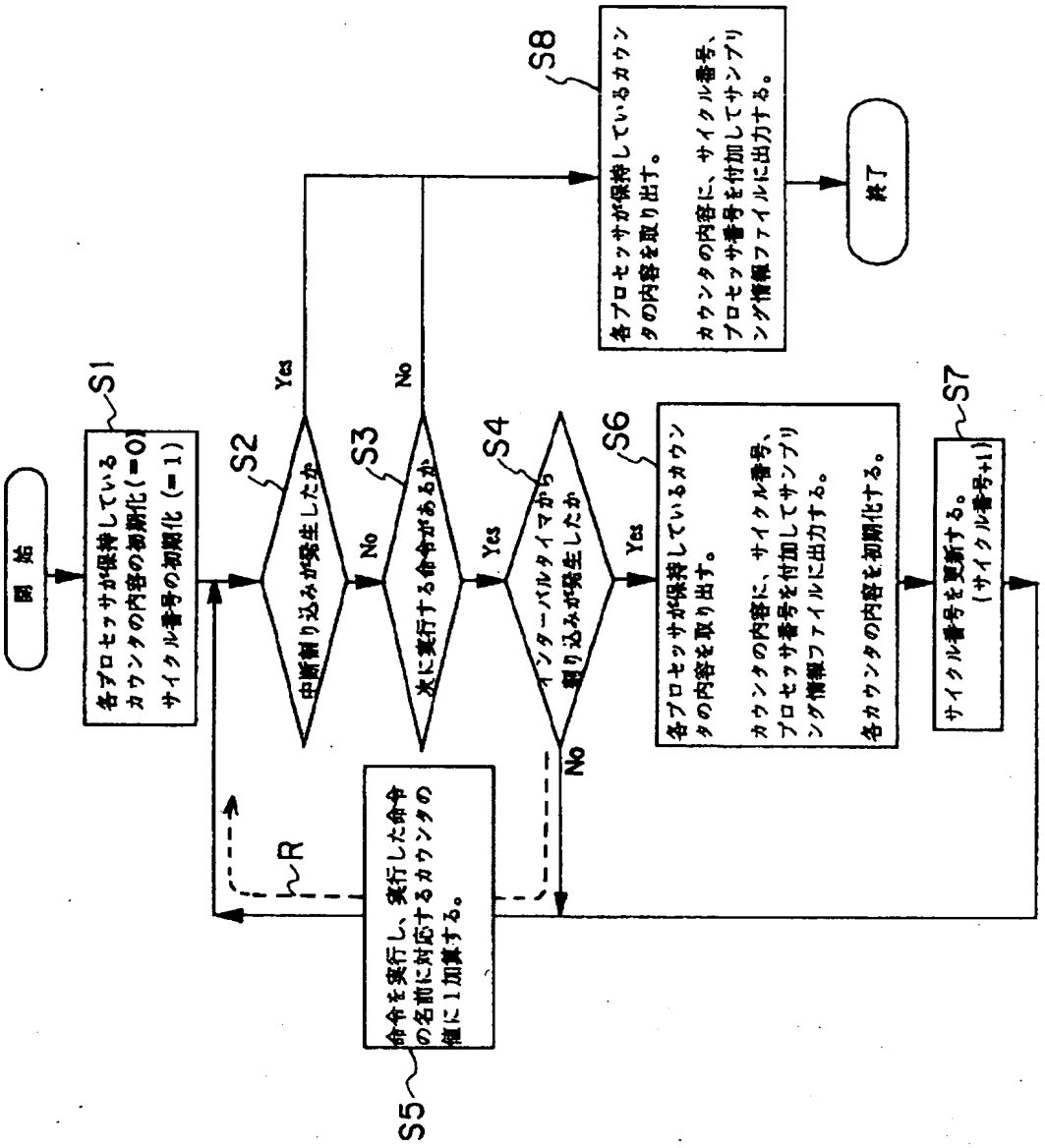
【図７】

各命令ごとの実行状況とプログラム全体の実行状況を示す図



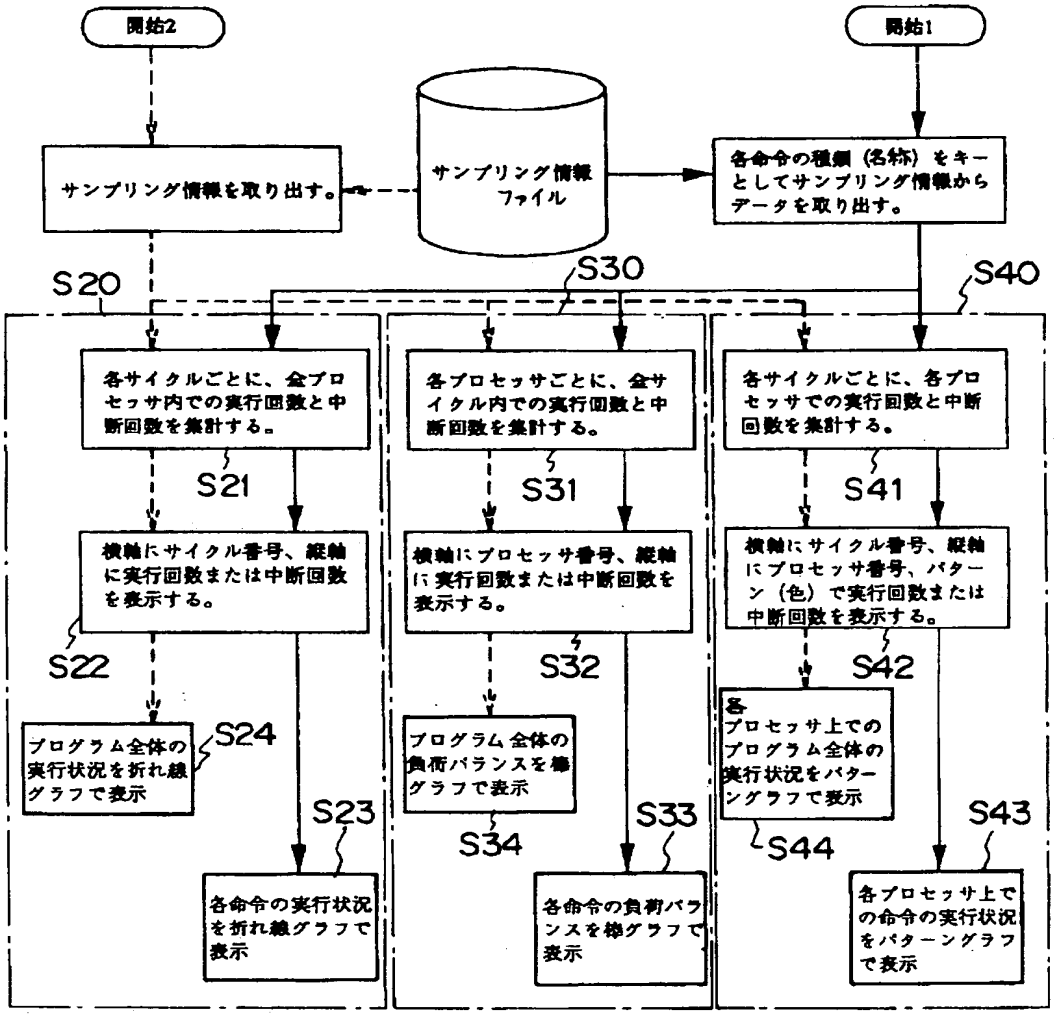
【図５】

サンプリング装置における情報収集処理のフローチャート



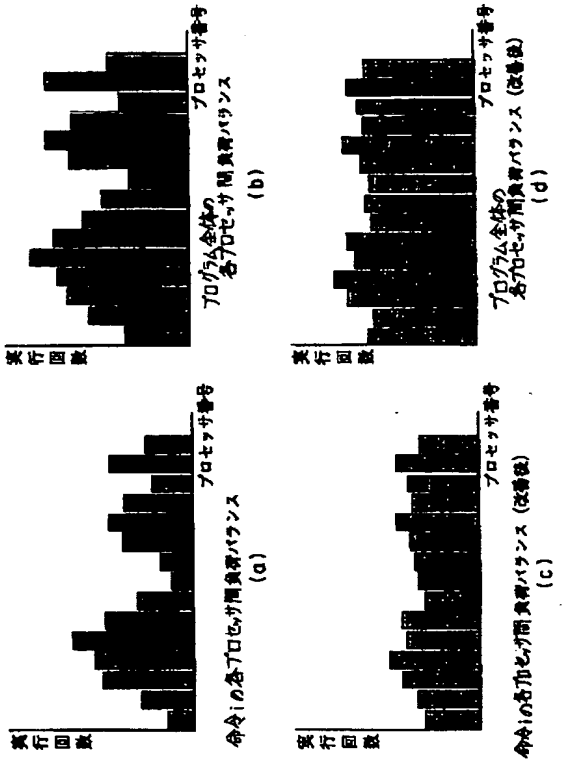
【図６】

編集／表示部における編集表示処理のフローチャート



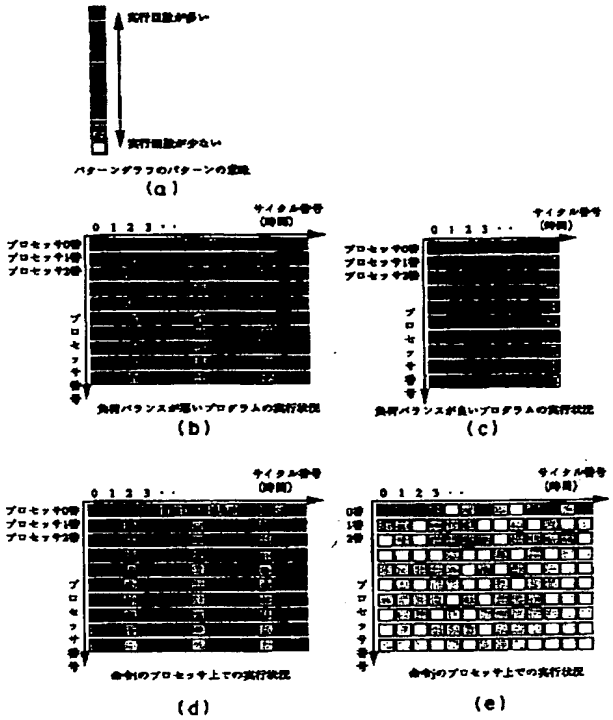
【図８】

各命令ごとの各プロセッサ間負荷バランスとプログラム全体の各プロセッサ間負荷バランスを示す図



【図９】

各プロセッサ上でのプログラムの実況状況を示す図



フロントページの続き

(72)発明者　松澤　史子
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内